EXHIBIT A

Microsoft Patent Pre-disclosure Document

Title of Invention: Summary-Detail Cube Architecture Using Horizontal Dimension Partitioning in A Large OLAP Application

Document Author(s): Adam E. C. Yeh

Introduction:

In a large scale enterprise implementation of an OLAP (On-Line Analytic Processing) application, large dimensions (i.e., dimension that have more than 500k members) are the most difficult component to cope with in terms of development and operation for a production system. Two significant factors that would influence the large-scale OLAP application design are the scalability for the application on the server-side and the usability for users using a client tool.

First, large dimensions do not scale well because the limitation on the available memory addressable space in a hardware platform (most commercial OLAP implementations require dimensions to be loaded to memory first to improve query-time performance). Second, users will not be able to navigate through thousands of dimension members in any commercially available clients to find these that are of interest to them; this is primarily due to the limitation on the client machine memory and CPU cycles and the inherent problems of presenting large number of selections to users. However, the growth of our internet business (MSN) and the reporting requirements constantly expose our OLAP based reporting implementation to these problems. In the past year, we have implemented a horizontal partitioning strategy for all our cubes to efficiently process our cubes on a daily basis, down from 14-17 hours a day to less than one hour. At the same time, we have proved that we can make Microsoft Analysis Service scale to more than 13 months' data on-line without degrading its performance because the high degree of aggregations and partitions in our cubes. The similar approach may be applied to tackle the explosion of large dimensions; however, Analysis Service (or other OLAP implementation) does not have any dimension based partitioning support. As a result, we have to find other way to resolve problems caused by large dimensions. We proposed and claimed a "**Summary-Detail Cube**" architecture to effectively mitigate the large dimension problems. In this architecture, the summary cube provides an overview for our business on a higher level (domain level and up). The detail cubes (service level based), on the other hand, enable our users to drill down to directory and page level without having to mingle with the large number of records on these levels. This idea is analogous to navigating through a digital map: Users will navigate from overview map to detail map when locating their interested address. They want to "zoom in" to the detail map with focus on the proximity of the interested address.

This architecture enables us to scale as many members as we want on the page and directory levels because they are horizontally partitioned based on the service they are associated, not the entire collection of the directories and pages for all our networks. These detail cubes can be located on the same server as the summary cube or they can scale out to a different server or many of them for performance reasons. As the dimension members are horizontally partitioned, they are more navigable and usable from the UI perspectives. And, we are able to offer a more granular set of reports through our OLAP cube without incurring performance degradation. One of the most important design points we employed using this architecture is that when users perform a detail-level analysis, they are not interested in data that are not in their drill-path; therefore, it is unnecessary to present data that are out of focus to them during such an analysis.

In this disclosure form, we claim this invention to include the following items:

1. Summary-detail cube architecture
2. XML driven work-flow for cloning detail cubes
3. XML metadata driven UI zoom in/out functionality using Summary-detail cubes

This invention offers a state-of-the-art architecture, an xml driven work-flow, and a client side zoom in/out navigation for implementing large scale OLAP solution in enterprises using our current business intelligence technologies: Microsoft Analysis Services and Office Web Component, as the foundations. It solves the scalability problems caused by large dimensions by using the best features of Microsoft Analysis Services (AS) and Office Web Component (OWC) and glue them in a seamless way with "Summary-detail cube" architecture. If we can build a native support of this architecture to provide a better synergy for both AS and OWC, we will be able to implement any large scale of OLAP applications required by large enterprises. This will give Microsoft an advantage in the market place when competing against Oracle (Express), IBM, Hyperion Solutions, MicroStrategy in the enterprises space for the Business Intelligence (BI) applications. Furthermore, this architecture will also provides a framework for Microsoft productions that contain BI applications, such as Commerce Server, SABLE, etc.

Background

Our group, iDSS, is responsible for providing usage reporting services for entire MSN Networks. The average number of hits on the entire MSN Networks is close to 1.5 billions/day. The processing of this large amount of data has created a challenge to the scalability of our OLAP reporting implementation. In the past year, we have implemented horizontal partitioning strategies on our OLAP cubes (i.e. cubes are the multi-dimensional storage for data stored in OLAP to facilitate fast retrieval) to improve scalability and processing time. However, the increased number of services (e.g., eShop, Carpoint, Encarta, etc.) and their growing businesses have resulted in large number of physical entities (e.g., html pages, web site directories, etc.) that need reporting and trending the historical data for contractual requirements and competitive advantages. The size of these physical entities (e.g., html pages) in the entire MSN Networks over time has accumulated faster than any OLAP products and UI tools can handle, thus preventing them from performing satisfactorily on the server and client sides. There performance and usability problems are generally caused by the large dimensions. Unfortunately, most commercial OLAP implementations do not support any dimension partitioning strategies to mitigate problems incurred due to large dimensions. We understand the needs of reporting the entire MSN Networks businesses in a coherent way on the daily basis. However, we also have pressing needs to enable site specific groups to review their business performance on a detail level, such as the page level. With these two requirements in mind, we proposed the "Summary-detail cube" architecture to effectively address the large dimension problems by horizontally partitioning them. And at the same time, we devise an xml template driven workflow to clone the detail cubes so a detail cube can be created in an autonomous way. To enable the seamless integration of the summary level reports and the detail level reports, we also offer the idea of zoom in/out navigation for the XML based UI implementation.

The Invention

Our invention we claim is driven by the idea of providing different levels of granularities in summary and detail cubes to improve the scalability and the usability of the application. It consists of the following components:

Summary-detail Cube Architecture Using Horizontally Partitioned Dimension

In a typical OLAP application, dimensions are defined to allow users to look at the facts (metrics) from different perspectives. When one of the dimensions grows unmanageable, users will have different time using it to slice and dice the metrics for analysis. It also causes the performance problem during aggregations as it triggers larger degree of data explosion due to the increased number of permutations that the system needs to calculate and the amount of physical memory it needs when loading the dimensions. In our case, we have a dimension called "Target," which provides the physical taxonomy for our web usage analysis. It consists of the following levels and their respective volume after the database has been on-line for 16 months:

Level Name	*Number of Members*	*Example*
Network	7	CCG Network, OEM, etc.
Service	66	CarPoint, eShop, etc.
Site	592	Encarta English, etc.
Domain	1911	www.msn.com, etc.
Directory/Subdirectory	1,237,443	migrate.msn.com/htm, etc.
Page	4,200,513	topic.asp, etc.

With this large amount of data on directory and page levels, users will have difficult time navigating through these dimension members when slicing and dicing the metrics. However, it is also important to provide a summary level of web usage to our users so they know the general performance of the entire MSN Networks. We also recognize that directory/page level information is more relevant to a service or a site as they are physically correlated; site property owners are the users who will drill down the directory/page level information for their specific analysis. For example, it's very unlikely for CarPoint analysts to drill down on Encarta's directory and page level information as they are not directly related to CarPoint's performance. However, in a typical OLAP application, pages in CarPoint unavoidably co-exist with these in Encarta as they come from the same dimension. We can image that if the OLAP application has to deal with more than 4 million pages when a user drills down to the page level, the application will simply fail to perform satisfactorily due to the volume of the dimension. We therefore propose the "Summary detail cube architecture" to segregate the "Target" dimension into different partitions using service level as the key. For example, CarPoint will have its directory/page level information available on its own detail cube, instead of having to share it with other services. In Figure 1 (see the next section), we show the "Summary detail cube implementation" for our NetMetrix OLAP application. NetMetrix is now a production application that offers the web usage reporting to our MSN Networks businesses. The idea of "Summary detail cube architecture using horizontally partitioned dimension" can be described with the implementation of NetMetrix cubes. In the NetMetrix summary cube, it contains two groups of physical cubes: cubes for the additive metrics (e.g., page views) and cubes for the non-additive metrics (unique user counts). They are virtualized to a logical cube, called "NetMetrix." Detail cubes have exactly the same dimensionality as the NetMetrix summary cube, except their "Target" dimensions are partitioned and include the directory/page level dimension members. For example, CarpointNetMetrix is the detail cube for CarPoint service; it has exactly the same dimensions as NetMetrix summary cube. However, its "Target" dimension only contains the sub-tree of the "CarPoint" service, extending to the directory/page levels.

XML templates driven work-flow for cloning detail cubes

To ensure manageability of "Summary detail cube architecture," we also devise an XML template driven work-flow implementation to clone detail cubes. In the context of NetMetrix OLAP application, we need to add the detail cubes if a new service is being added to the MSN Networks. The idea is to capture the metadata information about the dimension partitioning strategy and detail cubes in a set of XML files. By setting the partition key (in this case, the new service name), the workflow will execute to create the detail cubes and the partitioned dimension associated with them. The creation of these XML metadata is carried

out by parsing the OLAP services' object model to the XML elements (in our case, it's the Decision Support Object for Microsoft Analysis Service 2000). These XML metadata files then are revised to two templates: one for the partitioned "Target" dimension; the other is for the detail virtual cube and its physical cubes. The user only needs to decide which service detail cube to clone and the work-flow will perform the following steps to deploy all necessary objects:

1. Create a database view to horizontally partition the "Target" dimension, extending from domain to directory/page levels.
2. Use the XML templates and the new service name to create the XML metadata for the OLAP dimension and cubes to be created.
3. Apply a COM based application to read the XML metadata and deploy the OLAP objects to the OLAP services (in this case, the DSO application and Microsoft Analysis Services 2000).

These steps are automated in a workflow implementation using Microsoft Data Transformation Services. And Figure 2 depicts the processes of the work-flow.

XML metadata driven UI zoom in/out functionality using Summary-detail cubes

The usability of the "Summary detail cube" architecture is further enhanced by the client side zoom in/out functionality. We coined this functionality as zoom in/out because it is analogous to navigating through different scales of digital maps or focusing on the targeted objects when taking pictures. As the summary and detail cubes have identical dimensionality, users can navigate from the summary cube to the detail cube seamlessly. This functionality is enabled by a XML metadata file configured to identify the behavior of the zoom in/out events. They are defined as follows:

Zoom In:
Drill through to detailed cubes with the same dimensionality, preserving selections and filters.

Zoom Out:
Roll up to summary cubes with the same dimensionality, preserving selections and filters.

These events are enabled based on the context in which the users interact and their definitions are stored in a XML metadata. When a new action is added, there is no coding required and the user only needs to add new entry for the action in the XML metadata. To certain extent, traditional drill-across and drill-through events can also be implemented in such a manner. Figure 3 describes examples of zoom in/out and drill-across implementations.

The novelty of this implementation is the idea of zooming in from summary to detail and zooming out from detail cube to summary cube. Most drilling-down/roll-up implementations do not support pointing to different physical cubes so their dimensions can be dealt with separately; as a result, their performance will be constrained by the number of records existing in dimensions.

Extension of the Invention and Possible Alternative Implementations

Many-to-many relationship
The "Summary-detail cube" architecture can have more than one dimension being horizontally partitioned. For example, we can partition a second dimension to create a two dimension detail cube grid. On the other hand, we can also implement a second summary cube if necessary for different roll-up. In fact, their structure can be many-to-many relationship in the summary-detail cube architecture.

Implied vertical partitioning
The idea of detail cubes implementation also implies "vertical partitioning" strategies for the dimension as it enables more "attributes/levels" to be included in that dimension.

Different system configurations
The summary and detail cubes can be deployed on different servers. This architecture offers the flexibility of putting detailed cubes along with the summary cube or separating them into a different server. There are the following configurations we can do in this architecture:

1. Detailed cubes and summary cube are deployed on the same server and the same OLAP database: this configuration enables us to share the common dimensions between them. For example, summary cube always shares the same dimensionality as the detailed cubes. The only difference between the summary cube and detailed cubes is the "horizontally partitioned" dimension.
2. Detailed cubes can be deployed on another server to increase scalability.
3. Groups of detailed cubes can be deployed to different servers to offer the best scalability.

Possible middle-tier implementation for zoom in/out
We implemented the zoom in/out functionality using client side application (in this case, the Office Web Component). However, we expect this kind of functionality can also be implemented on middle-tier to offer better reference-ability among data from different data stores and better scalability for query-time performance.

Summary

The implementation of the "Summary-detail cube" architecture for NetMetrix OLAP application has enabled iDSS to effectively present reports for entire MSN Networks, in spite of the presence of the large dimensions. This architecture also enables iDSS to provide OLAP reporting services from summary to detail levels without incurring performance penalty. In summary, we solve the following problems associated with large dimensions:

1. Scalability: summary-detail cube architecture partitions large dimensions horizontally and therefore the implementation can predict a linear scaling performance.
2. Usability: the zoom in/out functionality facilitates the natural drill-path for analyzing detailed information in a summary-detail cube architecture. Users can navigate easily and find the interested dimensions quickly for analysis.
3. Manageability: the XML template driven workflow implementation automates the cloning of the detail cubes, which reduces the potential overhead for maintaining the OLAP application when additional detail cubes need to be added.

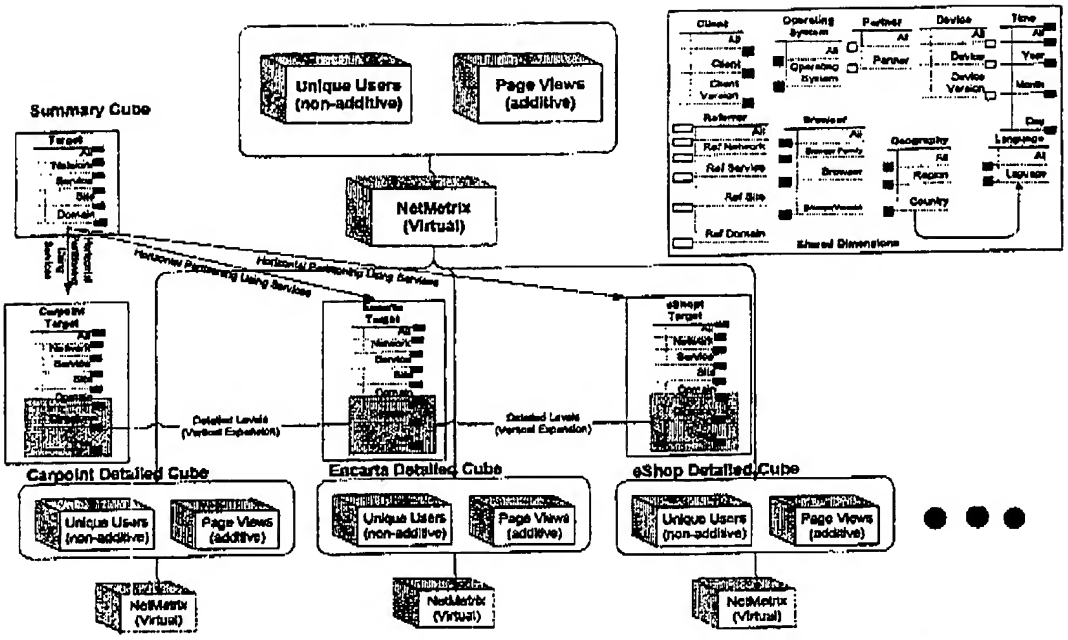

Figure 1 Summary-detail Cube Architecture (in The Context of MSN iDSS NetMetrix Application)

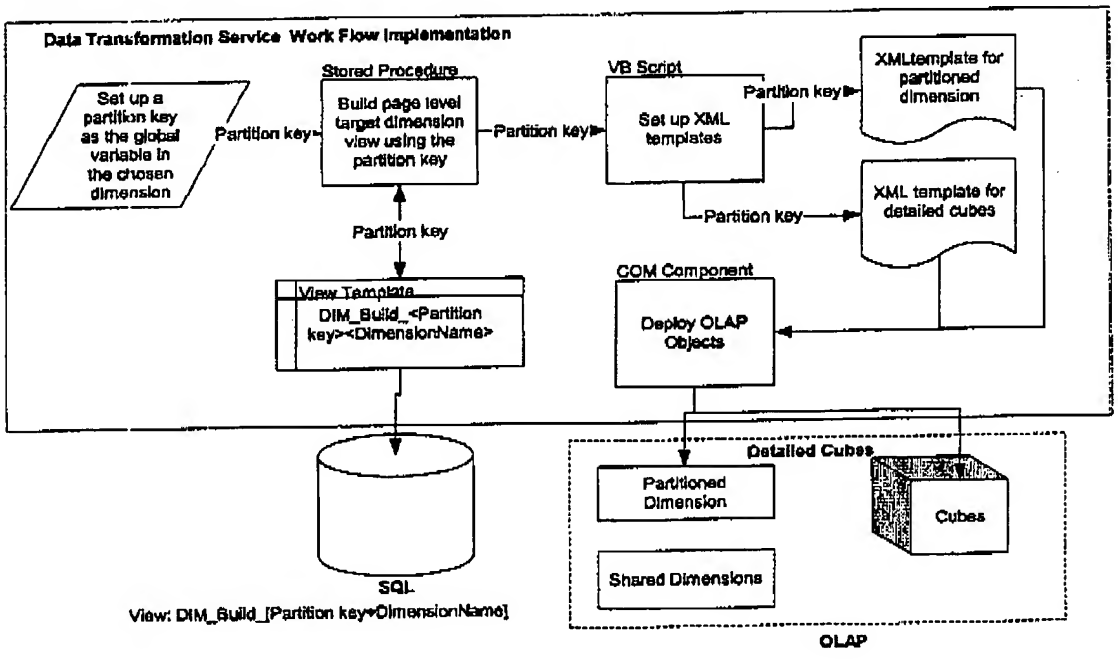

Figure 2 Work-flow for Cloning A Detail Cube

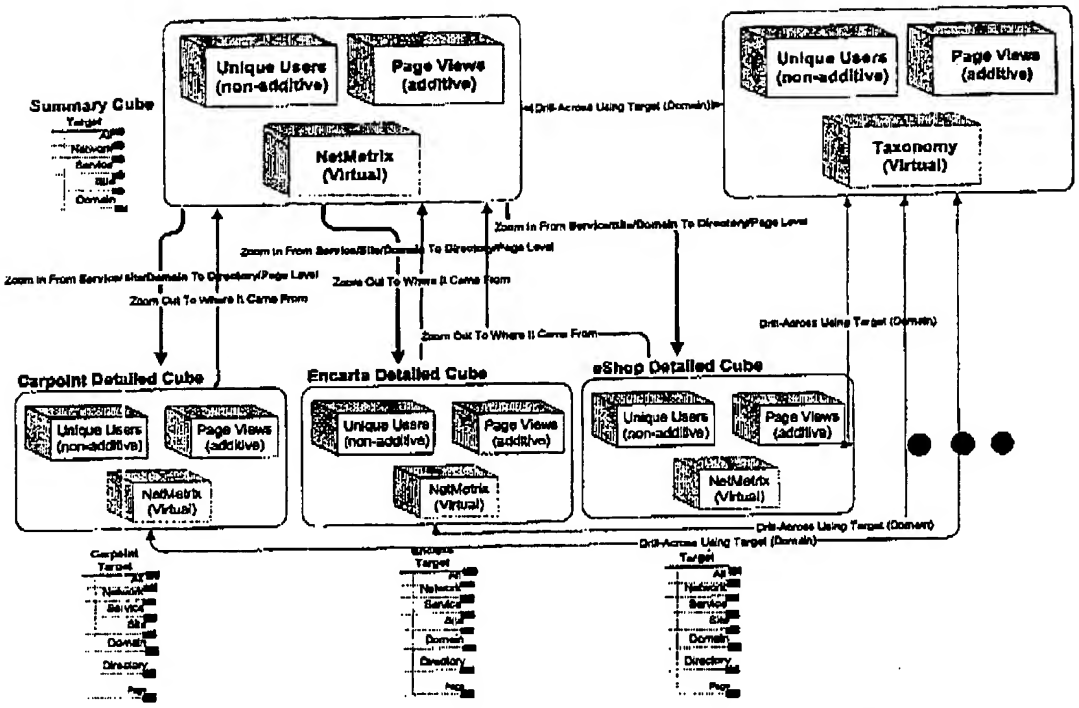

Figure 3 Zoom In/Out And Drill-Across Using Summary-detail Cubes